

ЛАБОРАТОРНА РОБОТА 4

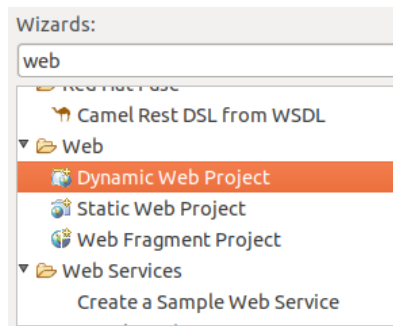
Тема: Реалізація додатків з використанням засобів рівня представлення

Мета: Створення web-проекту для моніторингу визначених ресурсів

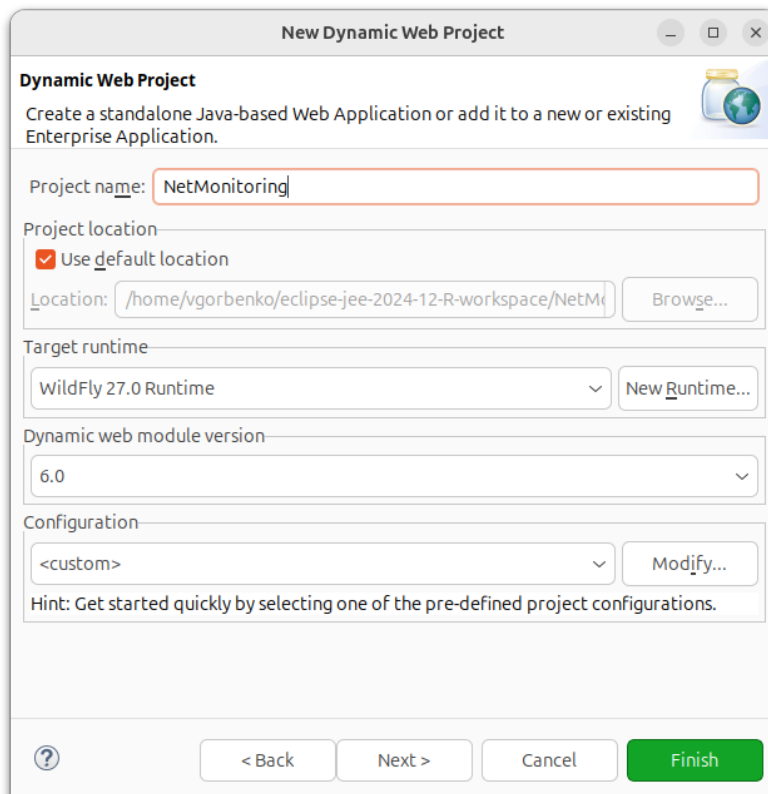
ПРАКТИЧНА ЧАСТИНА

Для розробки проекту припустимо застосування корпоративних технологій Java.

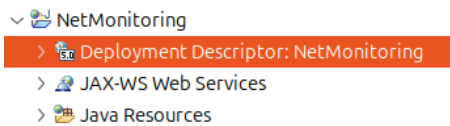
1. Створюємо динамічний web-проекту в IDE Eclipse. Для чого необхідно перейти до меню-візарда проектів: File -> New -> Other. В якості тексту пошуку можна набрати Web, що значно прискорить перехід до необхідного типу проектів. Обираємо Dynamic Web Project:



Наступним буде вікно для створення Dynamic Web Project:

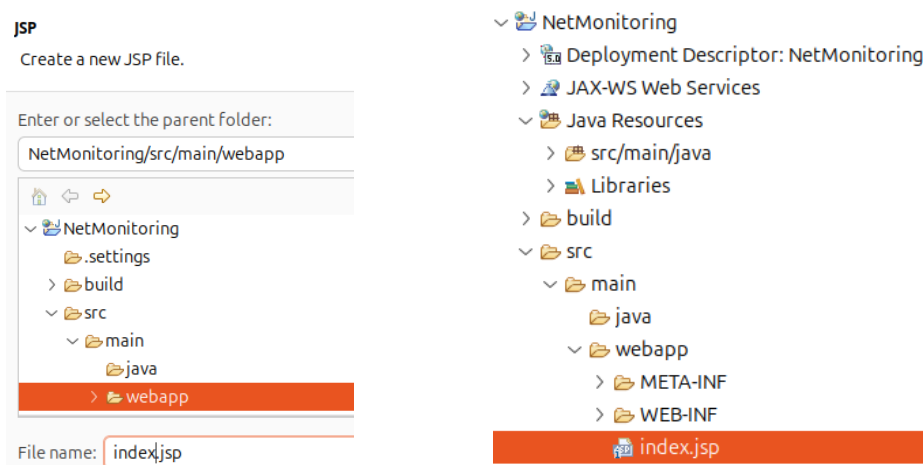


Задаємо ім'я проекту, наприклад, NetMonitoring. Для сумісності з обраною версією WildFly встановлюємо значення «Dynamic web module version» - 6.0. Після натискання Finish утворюється структура нового проекту. Упевніться, що в структурі є «Deployment Descriptor»:



Файли Java будуть знаходитись у теці Java Resources/src.

2. Для створення jsp-файлу необхідно на теці з іменем проекту клацнути правою кнопкою миші і обрати New -> JSP File. Для обробки файлу за замовчуванням задаємо йому ім'я index.jsp.



Замінімо автоматично створений код файлу на наступний:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Monitoring Net</title>
</head>
<%String errMsg = null; String testMsg = null;%>
<%if ("POST".equalsIgnoreCase(request.getMethod()) &&
    request.getParameter("submit") != null) {%>
    <jsp:useBean id="testNet" class="ua.edu.znu.netmonitor.TestNet">
    <jsp:setProperty name="testNet" property="*" />
    </jsp:useBean>
    <%
if("URL".equalsIgnoreCase(testNet.getTest())) testMsg = testNet.doTestURL();
if("IP".equalsIgnoreCase(testNet.getTest())) testMsg = testNet.doTestIP();
    errMsg = testNet.getTest();
    %>
    %>
</if>
</if>
<body>
<h2>Test</h2>
<%if (errMsg != null) { %>
    <span style="color: red;"><%out.print(errMsg); %></span><br>
    %>
<%if (testMsg != null) { %>
    <span style="color: blue;"><%out.print(testMsg); %></span><br>
    %>
<form method="post">
    Type of test:<br>
```

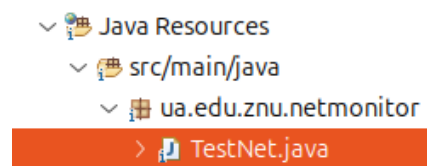
```

        <div>
            <input type="radio" id="testURL" name="test" value="URL" checked>URL
        </div>
        <div>
            <input type="radio" id="testIP" name="test" value="IP">IP
        </div>
        <button type="submit" name="submit">Submit</button>
        <button type="reset">Reset</button>
    </form>
</body>
</html>

```

Файл забезпечує простий інтерфейс для створення запиту на тестування ресурсів мережі та відображення результатів.

3. Додамо до проекту клас JavaBeans, що буде виконувати бізнес-логіку проекту:



та змінюємо його вміст:

```

package ua.edu.znu.netmonitor;

import java.net.HttpURLConnection;
import java.net.SocketTimeoutException;
import java.net.URL;
import java.net.UnknownHostException;

public class TestNet {
    private boolean testURL;
    private boolean testIP;
    private String test;

    public String getTest() {
        return test;
    }

    public void setTest(String test) {
        this.test = test;
    }

    public boolean getTestURL() {
        return testURL;
    }

    public boolean getTestIP() {
        return testIP;
    }

    public void setTestURL(boolean testURL) {
        this.testURL = testURL;
    }
}

```

```

public void setTestIP(boolean testIP) {
    this.testIP = testIP;
}

public String doTestURL() {
    String resultTest = "<br>";
    for (int ipCount = 0; ipCount < ipAddresses.length; ipCount++)
    {
        final int index = ipCount;
        resultTest += index + "<br>";
        resultTest += ipAddresses[index][0]+" "+ipAddresses[index][1]+" "+<br>";
        resultTest += testURL(
            ipAddresses[index][0],
            ipAddresses[index][1],
            ipAddresses[index][2]);
    }
    return resultTest;
}

public String doTestIP() {
    String resultTest = "<br>";
    for (int ipCount = 0; ipCount < ipAddresses.length; ipCount++)
    {
        final int index = ipCount;
        resultTest += index + "<br>";
        resultTest += ipAddresses2[index][0]+" "+ipAddresses2[index][1]+" "+<br>";
        resultTest += testURL(
            ipAddresses2[index][0],
            ipAddresses2[index][1],
            ipAddresses2[index][2]);
    }
    return resultTest;
}

private String testURL(String ipType, String url, String timeout)
{
    String resultTest = "";
    String result = "";
    int processReturnValue = 0;
    boolean wasLastStatusError = false;
    HttpURLConnection connection = null;
    int code = -1;

    String ip = url.startsWith("http://") ? url : ("http://" + url);

    try
    {
        URL siteURL = new URL(ip);
        connection = (HttpURLConnection) siteURL.openConnection();
        connection.setRequestMethod("GET");
        connection.connect();
        code = connection.getResponseCode();

        if (code >= 400)
            processReturnValue = 1;

        result = String.valueOf(code) + ": "
            + connection.getResponseMessage();
    }
    catch (UnknownHostException | SocketTimeoutException e)
    {
        result = e.toString();
        processReturnValue = 1;
    }
    catch (Exception e)
    {
        result = e.toString();
        processReturnValue = -1;
    }
    finally
    {
        if (connection != null)
        {
            connection.disconnect();
            connection = null;
        }
    }
}

```

```

switch (processReturnValue)
{
    case 0: // No error
        resultTest += "Network Connection Established to " + ipType
+ " Network (" + url + ")" + result + "<br>";
        break;
    case 1: // Connection error
        if (!wasLastStatusError)
        {
            resultTest += "Network Connection Error to " + ipType
+ " Network (" + url + ")" + result + "<br>";
            wasLastStatusError = true;
        }
        break;
    case -1: // Configuration or programming error
        if (!wasLastStatusError)
        {
            resultTest += "Configuration or Programming Error to "
+ ipType + " Network (" + url + ")" + result + "<br>";
            wasLastStatusError = true;
        }
        break;
} // end case

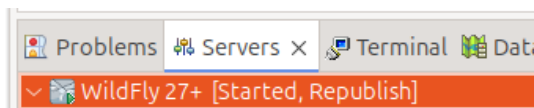
return resultTest;
}

private static String[][] ipAddresses =
{
    {
        "Open", "www.google.com", "3000"
    },
    {
        "VPN", "znu.edu.ua", "5000"
    }
};
private static String[][] ipAddresses2 =
{
    {
        "Open", "216.58.214.228", "3000"
    },
    {
        "Open", "192.168.0.1", "5000"
    }
};
}

```

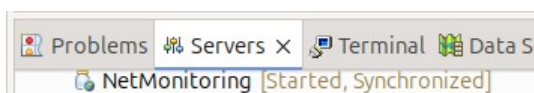
4. Запуск на WildFly

Для того, щоб запустити створену сторінку у web-браузері необхідно розгорнути додаток у контейнері сервлету, який реалізовано на сервері додатків, наприклад, WildFly. Для цього необхідно запустити WildFly:

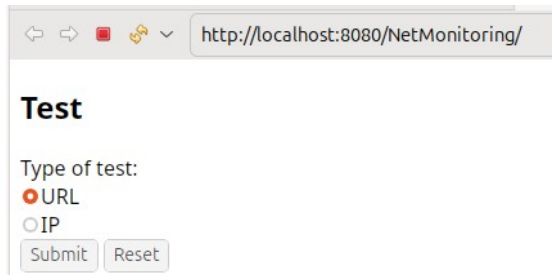


Для розгортання додатку необхідно клацнути правою кнопкою на WildFly у вікні Servers та обрати пункт Add and Remove. У діалоговому вікні, що з'явиться, обираємо проект для розгортання на сервері та натискаємо Finish.

Після цього сервер додатків повинен розгорнути та запустити додаток.



Для перевірки роботи додатку відкриємо його у браузері за відповідною адресою <http://localhost:8080/NetMonitoring/>



Якщо обрати тип тестування ресурсів мережі та натиснути кнопку Submit, то у відповідь отримаємо, наприклад:

Test

URL

0

Open www.google.com

Network Connection Established to Open Network (www.google.com)200: OK

1

VPN znu.edu.ua

Network Connection Established to VPN Network (znu.edu.ua)301: Moved Permanently

Type of test:

URL

IP

Submit

Reset

Завдання

1. Реалізуйте представлений проект та виконайте тестування мережі за типом “URL” та “IP”. Зробіть скриншоти виконаних завдань.
2. Поясніть принцип роботи проекту.
3. Знайдіть в наданому коді необроблені винятки, умовні перевірки, проблеми з ресурсами. Реалізуйте покращення коду, наприклад, використанням try-catch для обробки виключень та try-with-resources для коректного закриття з’єднань.
4. Змініть реалізацію проекту на використання технології сервлетів. Доповніть проект формою, за допомогою якої будуть задаватись ресурси для моніторингу.
5. Реалізуйте збереження історії перевірок. Додайте сторінку для перегляду історії (дата, тип тесту, результат).
6. Підготуйте звіт, в який додайте скриншоти виконаних прикладів та код рішення поставлених задач.